

EM 044522632

EL 685270532

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

System and Method for Software Licensing

Inventor(s):

Pradyunma K. Misra

Bradley J. Graziadio

Terence Spies

ATTORNEY'S DOCKET NO. MS1-197US

DOCKET # 20040200

Insert
A i

1

2

5

A 6

12

18

24

1 been installed, whether the copies have expired, whether they need upgrading, and
2 so forth.

3 Accordingly, there is a need to develop a new approach to licensing
4 software in a manner that assures that the terms are being met and assists the
5 licensee with monitoring whether it is in compliance with the software license.

6 7 SUMMARY

8 This invention concerns a system and method for licensing software. The
9 system and method provides confidence to the vendor that the software license is
10 being complied with, while also assisting the purchaser in monitoring its own
11 compliance with the license.

12 According to one aspect of this invention, computer software licenses are
13 electronically issued as digital certificates that can be distributed in one-to-one
14 correlation with individual client computers and traced to an issuing authority.

15 According to another aspect, the system includes a license generator located
16 at a licensing clearinghouse and at least one license server and multiple clients
17 located at or affiliated with a company or other entity. Because the clients might
18 not have network connectivity to the license server, one or more intermediate
19 servers may act as an intermediary for the clients. These intermediate servers are
20 otherwise common servers that provide resources to clients, but with the added
21 ability to facilitate connectivity to the license server for purposes of distributing
22 software licenses to the clients.

23 When a company wants a software license, it sends a purchase request (and
24 an appropriate fee) to the licensing clearinghouse. The license generator at the
25 licensing clearinghouse creates a license pack containing a set of one or more

1 the license server issues the existing license to the client. This is actually reissuing
2 of the same license that was previously issued. This allows the client to gracefully
3 recover licenses when they are lost.

4 In one implementation, the license server determines an appropriate type of
5 license based in part on the client's operating system platform. The license server
6 derives the platform information by establishing a trust relationship with the client
7 and then querying its platform type. If a software license is available for
8 allocation, the license server grants a software license that is appropriate for the
9 client's platform.

10 To prevent an issued license from being copied from one client machine to
11 another, the software license is assigned to a specific client by including its client
12 ID within the license. The software license also has a corresponding license ID
13 that is associated with the client ID in a database record kept at the license server.

14 The license server digitally signs the software license. The license is passed
15 to the client, where it is stored in a local cache at the client. Once a client has
16 obtained a license, it is responsible for managing the storage of that license.

18 **BRIEF DESCRIPTION OF THE DRAWINGS**

19 The same reference numbers are used throughout the drawings to reference
20 like components and features.

21 Fig. 1 shows a software licensing system.

22 Fig. 2 shows a block diagram of a computer used to implement the software
23 licensing system.

24 Fig. 3 shows a functional block diagram showing software components and
25 databases that implement the software licensing system.

1 Fig. 4 shows steps in a method for issuing a license pack of individual
2 licenses.

3 Fig. 5 shows steps in a method for initiating a connection between a client
4 and a server and determining whether the client has a valid license.

5 Fig. 6 shows steps in a method for distributing a software license to a client.

6 Fig. 7 shows steps in a method for challenging a client prior to granting a
7 software license to that client.

8 Fig. 8 shows steps in a method for upgrading a software license.

9
10 **DETAILED DESCRIPTION**

11 The following discussion assumes that the reader is familiar with public key
12 cryptography. For a basic introduction to cryptography, the reader is directed to a
13 text written by Bruce Schneier and entitled, "Applied Cryptography: Protocols,
14 Algorithms, and Source Code in C," published by John Wiley & Sons, copyright
15 1994 (second edition 1996), which is hereby incorporated by reference.

16 Fig. 1 shows a system 20 for licensing software. The system 20 has a
17 licensing clearinghouse 22 that creates and issues valid software licenses to one or
18 more companies, firms, agencies, or other entities, as represented by company 24.
19 The clearinghouse 22 is a separate entity from the company 24. Examples of the
20 clearinghouse include a software manufacturer, a software vendor, or a third party
21 agent that is authorized to issue software licenses on behalf of the software
22 manufacturer or vendor.

23 The company 24 contacts the clearinghouse 22 when it desires to purchase a
24 software license to run software on the company computers. The clearinghouse 22
25 has a license generator 26 that creates a "license pack" containing a set of one or

Exemplary Computer Used to Implement Servers and/or Client

The license generator 26, license server 28, and intermediate server 32 are preferably implemented as computer servers, such as Windows NT servers that run Windows NT server operating systems from Microsoft Corporation or UNIX-based servers. It is noted, however, that the license generator 26 and license server 28 may be implemented using other technologies, including mainframe technologies, as long as they share an inter-operable communication mechanism like remote procedure call (RPC) and these systems are secure.

The clients 30 can be implemented as many different kinds of computers, including a desktop personal computer, a workstation, a laptop computer, a notebook computer, a handheld PC, and so forth. The clients 30 may further represent a terminal device, which is a low cost machine with limited local processing and local memory. The terminal device includes a display, a keyboard, a mouse (optional), limited computer resources like memory, and enough intelligence to connect to an intermediate server. All applications run at the server. The terminal merely provides a connection point to the server-based processing.

The clients 30 might also represent a network-centric computer, such as a Network Computer (or NC) or a Net PC.

Fig. 2 shows an example implementation of a computer 40, which can be used to implement the license generator 26, license server 28, and intermediate server 32. The server 40 includes a processing unit 42, a system memory 44, and a system bus 46 that interconnects various system components, including the system memory 44 to the processing unit 42. The system bus 46 may be implemented as any one of several bus structures and using any of a variety of bus architectures, including a memory bus or memory controller, a peripheral bus, and a local bus.

1 of Windows. The operating system 70 may alternatively be other types, including
2 Macintosh and UNIX-based operating systems.

3 A user may enter commands and information into the computer 40 through
4 input devices such as a keyboard 78 and a mouse 80. Other input devices (not
5 shown) may include a microphone, joystick, game pad, satellite dish, scanner, or
6 the like. These and other input devices are connected to the processing unit 42
7 through a serial port interface 82 that is coupled to the system bus 46, but may
8 alternatively be connected by other interfaces, such as a parallel port, game port, or
9 a universal serial bus (USB).

10 A monitor 84 or other type of display device is also connected to the system
11 bus 46 via an interface, such as a video adapter 86. The computer 40 has a
12 network interface or adapter 88, a modem 90, or other means for establishing
13 communications over a network 92.

14 15 System Architecture

16 Fig. 3 shows an exemplary software/hardware architecture of the system
17 20. The architecture includes four components: a license generator 26, a license
18 server 28, a client 30, and an intermediate server 32. The license generator 26
19 produces license packs for a fee and the license server 28 consumes the licenses by
20 installing them. In turn, the license server 28 distributes a license to the client 30
21 with the help of the intermediate server 32. The client 30 then uses the license to
22 gain access to the resources provided by the intermediate server 32.

23 The entity or organization that owns, or is responsible for, the license server
24 28 registers itself with an independent certifying authority that is trusted by both
25 the organization and the clearinghouse. The organization submits information

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

Number	generator to prevent the license pack from being installed multiple times on the same license server.
Issue Date	The date the license pack is issued by the clearinghouse.
First Active Date	The date on which the licenses within the license pack can first be used.
Expiration Date	The date on which the licenses within the license pack will expire. A license could be set such that it does not expire.
Begin Serial Number	The beginning serial number for the licenses in the license pack. The number is used to assign a unique serial number to each license within the license pack.
Quantity of Licenses	The number of licenses contained within the license pack.
Number of Human Descriptions	The number of Human descriptions included for the license pack.
Array of Human Descriptions (Locale, Description)	<p>Locale—Identifies the locale for the Human Description.</p> <p>Human Description—A description of the contents of the license pack in a localized form.</p>
Manufacturer	Identity of the manufacturer of the product being licensed.

09/24/03 13:00

1 Manufacturer-Specific
2 Product Data

Manufacturer-dependent information used
to identify the product. As an example,
this data might include:

- 3 1. Product Family Code
- 4 2. Product Version
- 5 3. License Type

6 Signature

Digital signature generated by the license
generator using the clearinghouse private
key.

8
9 Clearinghouse's Public
10 Key Certificate

The certificate issued to the clearinghouse
and containing the clearinghouse's public
key. This public key is used to sign the
encrypted license pack.

11
12
13 One parameter of the purchase request and subsequent license pack is the
14 client platform type. As one possible implementation, the system 20 is configured
15 to reliably recognize four different platform types: Windows, Non-Windows,
16 Legacy, and Direct-Connect. A "Windows"-type platform means the client
17 computer runs a 32-bit version of Microsoft Windows operating system (e.g.,
18 Windows 95, Windows 98, Windows NT, etc.). A "Non-Windows"-type platform
19 means the client computer runs an operating system other than a Windows brand
20 operating system. A "Legacy"-type platform indicates that the client runs an older
21 version of an operating system that cannot be adequately determined by the license
22 server as a "Windows"-type or a "Non-Windows"-type. A "Direct-Connect"
23 platform means the client is a terminal that attaches directly to the server's bus and
24
25

thus, all of the operating system functionality is provided directly by the server.

Table 2 summarizes the platform types.

Table 2: Platform Types

<u>Platform Type</u>	<u>Description</u>
Windows	Authenticated client platforms that are Win32-based.
Non-Windows	Authenticated client platforms that are not Win32-based.
Legacy	Clients that are implemented with older operating systems that are incapable of fielding a client platform challenge from the license server. There is no way of determining whether or not the client's platform is Win32 capable.
Direct-Connect	Multi-console clients that are attached directly to the server's BUS. These clients derive the operating system capabilities from the server itself.

The license server 28 has a license pack installer 110 and a secure license store 112. The license pack installer 110 installs the license pack(s) 108 received from the license generator on the secure license store 112. The license pack installer 110 may also be used to order the license packs, when such purchase requests are made electronically.

The license pack is stored in a secured database. A library of routines for adding, removing, querying, upgrading and extracting licenses are used to manage the licenses within the license store. As noted above, the license packs are encrypted using the license server's private key to prevent users from tampering

Table 3: License Pack Table

<u>Field</u>	<u>Description</u>
License Pack ID	A unique identifier assigned by the license generator.
Quantity	The number of software licenses contained in the license pack.
Number Assigned	The number of software licenses that have been assigned to clients.
First Active Date	The date on which the licenses within the license pack can first be used.
Expiration Date	The date on which the software licenses in the license pack will expire.
Begin Serial Number	The beginning serial number for the licenses in the license pack. The number is used to assign a unique serial number to each license within the license pack.
Product-Specific Attributes	Product-dependent information to indicate specific features of a product. As an example, this date might include: <ul style="list-style-type: none"> 1. Product ID 2. Product Flags 3. Platform Type

The number assigned field need not be kept, but it helps eliminate the need to count the number of assigned licenses each time an administrator wants to determine how many free licenses are available.

The client assignment table 116 contains a list of all licenses that have ^{been} distributed to the clients. Each record in the client assignment table 116 is

assigned a unique license ID. The license ID serves two purposes: (1) it allows the table 116 to be indexed and (2) it provides a license tracking mechanism for the client. The client assignment table 116 also contains the license pack ID from which each license is derived.

Table 4 shows the fields in the client assignment table 116.

Table 4: Client Assignment Table

<u>Field</u>	<u>Description</u>
License ID	A unique identifier assigned by the license server to each software license, based on the begin serial number.
License Pack ID	The unique identifier assigned by the license generator.
Client ID	A unique identifier of the client to which the software license is granted.
Issue Date	The date on which the software license is issued to the client.

The license pack ID fields in the license pack table 114 and the client assignment table 116 can be used to join the tables in a one-to-many relationship; that is, one record identified in the license pack table 114 to many records in the client assignment table 116 as software licenses are issued to clients. This joiner yields a list of all software licenses assigned to clients from a given license pack. The client ID field enables the administrator to query all licenses for a particular client.

1 license server, and uses this public key to verify the client's signature before
2 installing the client image on the cache 120.

3 The license server 28 also has a request handler 122, a client authenticating
4 module 124, and a granting module 126. The request handler 122 receives
5 requests for software licenses from clients. The client request typically includes
6 the client ID. The request handler 122 passes the request to the client
7 authenticating module 124, which determines whether the client is authentic and
8 able to receive a software license.

9 As part of the authentication process, the client authenticating module 124
10 initiates a platform challenge requesting a client executable image from the client
11 30. One preferred approach to performing a platform challenge is described below
12 in more detail under the sub-heading "Platform Challenge".

13 The client authenticating module 124 compares the client executable image
14 received from the client to the client executable image stored in the client image
15 cache 120. The client is deemed authentic if the two images match. The client
16 authenticating module 124 informs the granting module 126 when the client is
17 authenticated.

18 The granting module 126 grants a software license from the secure license
19 store 112 to the authenticated client. To prevent an issued license from being
20 copied from machine to machine, the software license is assigned to a specific
21 client by assigning a client ID to the license and including that ID within the
22 license. The software license is also given a license ID. The license ID is
23 associated with the client ID in the client assignment table 116 to track which
24 client receives the issued license.

The license server 28, based on information derived from the license pack, fills in fields of a license data structure at the time the license is issued. As one example, the license data structure is implemented using an X.509 certificate, which is well known in the art. The license server 28 then digitally signs the software license using a signing key that is not disclosed to the client. Table 5 shows the data fields of a software license data structure.

Table 5: Software License Contents

<u>Field</u>	<u>Description / Purpose</u>
Version	Identifies the "data structure" version of the software license so newer licenses can be used on older servers.
License ID	A unique ID assigned to the software license by the license server at the time of issuance to the client.
Client ID	The unique identifier of the client to which the software license is assigned.
Issue Date	The date on which the software license is assigned to the client.
Expiration Date	The date on which the software licenses in the license pack will expire.
Product-Specific Attributes	Product-dependent information to indicate specific features of a product. As an example, this date might include: <ol style="list-style-type: none"> 1. Product ID 2. Product Flags 3. Platform Type
Signature	Digital signature generated by the license generator using the clearinghouse private key.

1 License Server's The license server's public key in certificate form,
2 Certificate as issued by the certifying authority.

3
4
5 As part of the granting process, the client assignment table 116 is updated to
6 reflect that a particular license having a specific license ID is issued to a particular
7 client having a specific client ID. Additionally, the number assigned field in the
8 license pack table 114 is updated to reflect that another license has been assigned
9 to a client.

10 The license pack installer 110, client image installer 118, request handler
11 122, client authenticating module 124, and granting module 126 are preferably
12 implemented as software programs executing on the license server 28. These
13 software programs are preferably implemented as part of the operating system at
14 the license server.

15 The intermediate server 32 acts as a go between for the client 30 and license
16 server 28. The intermediate server is a full-service server that is used regularly by
17 the client to perform normal tasks that are customary for the company or entity.
18 But, the intermediate server is further equipped with a client licensing unit 128 to
19 facilitate communication between the client 30 and license server 28. The
20 intermediate server 32 also has a legacy license store 130, which stores licenses for
21 clients whose platforms cannot generate a unique system ID.

22 The client 30 has a license requestor 132, a challenge handler 134, and a
23 license cache 136. The license requestor 132 initiates the license requests for
24 obtaining a software license from the license server 28. This involves connecting
25 to the intermediate server 32 and presenting a software license and a client ID to

Issuance of License Pack

Fig. 4 shows steps in a method for requesting and issuing a license pack from a license generator. At step 150, the license server 28 generates and sends a purchase request 106 to an authorized license generator 26. The request 106 contains information used by the license generator 26 to issue one or more software license packs to the requesting license server 28. The purchase request 106 contains the platform type (see Table 2), the quantity of licenses desired, the product ID, the license server's certificate (containing the license server's public key K_{LS_pub} and the license server ID), and the list of features that the license should enable. The license server can submit this information electronically to the license generator via the Internet, modem, e-mail, on a floppy diskette, or other electronic means. Additionally, the administrator at the company or entity might submit a purchase request to the licensing clearinghouse 22 in writing on paper, or place an order orally by telephone. The license server 28 typically submits a licensing fee with the purchase request, or sometime following the initial communication.

After collecting the fee for the software licenses, the license generator 26 creates a license pack containing a set of one or more individual software licenses and assigns a unique license pack ID to the license pack (step 152 in Fig. 4). The license generator 26 stores the collected information in the master license database 100 (step 154). The information from the license server 28 is correlated within the database 100 to the license pack ID. In this manner, the license pack ID is associated with a particular license server having a specific license server ID (step 156).

1 The license generator 26 encrypts the license pack of software licenses
2 using the license server's public key K_{LS_pub} , thus binding the license pack to the
3 requesting license server 28. The license generator 26 digitally signs the license
4 pack using its (i.e., the clearinghouse's) private signing key K_{CH_pri} (step 160 in
5 Fig. 4) and sends the license pack to the requesting license server 28.

6 The license pack 108 contains a set of one or more non-assigned licenses
7 and the license pack ID. Table 1 lists the contents of the license pack 108.

8 At step 164 in Fig. 4, the license server 28 uses the clearinghouse's public
9 signing key K_{CH_pub} to verify that the digital signature accompanying the license
10 pack 108 belongs to the license generator 26 of clearinghouse 22 and that the
11 license pack 108 has not been altered. If the signature is authentic and from a
12 known clearinghouse, the license server 28 decrypts the license pack contents
13 using its private key K_{LS_pri} (step 166). The license server 28 extracts the license
14 pack ID and queries the secure license store 112 to see if it already contains the
15 same license pack (step 168). If the license pack is new, the license server installs
16 it on the secure license store 112 (step 170).

17 18 Distribution of Licenses

19 Client Connection

20 Fig. 5 shows steps in a process that facilitates a client's initial connection to
21 the intermediate server. The client connects to the intermediate server 32 to ask
22 for services or data provided by the server. Prior to working with the client and
23 providing access to files, the intermediate server 32 wants to verify first that the
24 client has a valid software license issued by a recognized license server. The client
25 30 may or may not have a valid license, so the intermediate server makes an initial

1 evaluation when the client attempts to connect. Generally, if the client 30 has a
2 valid license, the client is permitted to connect and use the server's resources. If
3 the client 30 offers an invalid license, the client is disconnected. If the client 30
4 does not offer a valid license or offers an expired license, the intermediate server
5 32 facilitates the process of obtaining a new software license.

6 At step 172, the client 30 submits a connection request to the intermediate
7 server 32. The connection request includes the client's system ID that uniquely
8 identifies the computer. In response, the intermediate server 32 passes a list of the
9 product IDs required (step 174). In this manner, the intermediate server 32 limits
10 its acceptance of software licenses to those that are issued by legitimate and
11 authorized license servers.

12 With this information, the client 30 queries its license cache 136 to search
13 for a suitable license from a license server that appears on the list (step 176 in Fig.
14 5). If a software license is found, the client 30 sends the software license to the
15 intermediate server 32 along with the client ID; otherwise, the client 30 submits
16 only a client ID (step 178). The software license contains the digital signature of
17 the license server.

18 At step 180 in Fig. 5, the intermediate server 32 determines whether the
19 client submitted a software license. If so, the intermediate server 32 verifies
20 whether the digital signature belongs to an authorized license server and whether
21 the license contains a valid client ID (step 182). The client ID is checked by
22 extracting the client ID from the license (which was provided originally by the
23 licensing server, as described below) and comparing it to the client ID received
24 from the client. If the two match, the client ID passes.
25

1 If the digital signature or the client ID is not valid (i.e., the "not valid"
2 branch from step 182), the software license is deemed invalid. The client's request
3 for connection is then rejected and the client is disconnected. On the other hand, if
4 the digital signature and the client ID are both valid (i.e., the "valid" branch from
5 step 182), the intermediate server 32 checks if the license has expired (step 184),
6 the connection is completed if the license is still valid i.e. has not expired and the
7 client is allowed access to the services and files of the intermediate server (step
8 186).

9 In the event that the client 30 does not submit a valid license or submits an
10 expired license, the intermediate server requests a new software license from the
11 license server (step 188 in Fig. 5).

12 13 New License Grant

14 Software licenses are distributed to the client automatically by the license
15 server. As discussed above, when a client 30 connects to an intermediate server
16 32, the client must present a valid license. If it cannot, the intermediate server acts
17 as a proxy for the client and requests a license from its associated license server.

18 Fig. 6 shows steps in a method for granting a new software license from the
19 license server 28 to the client 30. The method begins with step 188, which is the
20 same new license request discussed above with respect to step 188 of Fig. 5. The
21 new license request includes the client's system ID and the product ID. In
22 response to the request, the license server 28 initiates a client challenge to
23 determine who the client is and what platform it is running (step 190). In general,
24 this involves generating a challenge and sending it to the intermediate server 32
25

1 (step 192). The intermediate server 32 forwards the challenge to the client 30
2 (step 194).

3 At step 196 in Fig. 6, the client responds to the challenge in a manner that
4 provides trusted information about client, including the platform type and the
5 client's public key. The response is passed to the intermediate server 32, which
6 forwards it to the license server 28 (step 198).

7 At step 200 in Fig. 6, the license server determines whether the response is
8 proper, and hence, whether the client is authentic. If the client is authenticated
9 (i.e., the "yes" branch from step 200), the license server proceeds with granting a
10 software license. The license server 28 first queries the secure license store 112 to
11 determine if a license for that client has already been issued (step 202). This
12 procedure accommodates the case in which the client has lost its valid software
13 license. If a non-expired license is found, the license server 28 forwards it to the
14 client 30.

15 Otherwise, the license server 28 attempts to allocate a software license for
16 the client, assuming a non-assigned license still exists in the license pack. If a
17 license can be allocated, the license server 28 retrieves a software license that is
18 appropriate for the client's platform from the secure software store 112 and grants
19 the software license to the client (step 204 in Fig. 6). The license server 28 adds a
20 record to the client assignment table 116 and the corresponding number assigned
21 field is updated to reflect one additional allocation.

22 To prevent the software license from being copied from one client machine
23 to another, the software license is assigned to the specific client by including its
24 client ID within the license. The software license also has a corresponding license
A 25 ID that is associated with the client ID in the client assignment table 116 in the

1 secure license store 112 at the license server. The contents of the license are
2 described above in Table 5.

3 The license server 32 digitally signs the software license (step 206) and
4 encrypts it using the client's public key K_{C_pub} (step 208), thereby binding the
5 license to the client. The encrypted license is forwarded to the intermediate server
6 32, which passes it on to the client 30 and completes the connection (step 210). By
7 encrypting the license, the client or the license server need not trust the
8 intermediate server because the intermediate server cannot maliciously utilize or
9 modify the encrypted license. It also removes the risk of a rogue server
10 masquerading as intermediate server. At step 212, the client 30 decrypts the license
11 using the client's private key K_{C_pri} and stores the license in the license cache 136.

12 In the event that the client's response to the challenge is deemed improper
13 (i.e., the "no" branch from step 200), the license server returns a rejection notice
14 (step 214 in Fig. 6). This rejection notice is passed on by the intermediate server
15 32 (step 216) and used to inform the user (step 218).

16 17 Platform Challenge

18 Fig. 7 shows a more detailed method for providing a platform challenge to
19 the client. In this illustration, the intermediate server 32 is shown as the go
20 between, with the forwarding steps omitted for ease of description.

21 An aspect of platform validation is establishing the authenticity of the
22 client. The system utilizes the client's executable image to generate a digital
23 signature that uniquely identifies the client. As noted above, the client's
24 executable image is available to the license server 28 because it is stored in the
25 client image cache 120.

1 When a client requests a software license from the license server, the client
2 30 submits a client software ID (step 220 in Fig. 7). The software ID is assigned
3 by the software manufacturer/vendor to be unique for each client release. The
4 client software ID is a bit field that contains a platform identifier, a vendor
5 identifier, and a client revision field. The arrangement of the bits depends on how
6 many platforms and clients are supported.

7 At step 222, the license server 28 uses the software ID to lookup the client's
8 executable image in the client image cache 120. If the image is not present in the
9 cache (i.e., the "no" branch from step 222), the client is denied a software license
10 and a rejection is returned to the client and informs the user (steps 224 and 226).

11 On the other hand, if an image is present (i.e., the "yes" branch from step
12 222), the license server 28 sends a challenge to the client 30 to establish a trust
13 relationship with the client (step 228). The challenge is preferably a 128-bit
14 random number.

15 The client 30 applies a one-way function to a combination of the challenge
16 and the client's image (step 230). Preferably, the client concatenates the challenge
17 and the client image and computes a hash value, as follows:

18
19
$$\text{Challenge Response} = \text{Hash}(\text{challenge}|\text{client image}|\text{challenge})$$

20

21 The client 30 sends the challenge response (i.e., the hash value) back to the
22 license server 28 (step 232).

23 Meanwhile, the license server 28 uses the software ID to retrieve a
24 reference copy of the client image from its cache 120 (step 234 in Fig. 7). The
25 license server then computes a test hash value using the same hash function, and a

concatenated version of the same 128-bit challenge and the client image retrieved from the cache 120 (step 236).

The license server 28 compares the test hash value (H') with the hash value (H) returned from the client (step 238). If the two values are the same, the client's platform information is extracted from the client software ID and a trust relationship established (i.e., the "yes" branch from step 238). Otherwise, the client is denied a software license and a rejection is returned to the client (i.e., the "no" branch from step 238).

Upgrading Licenses

The process for upgrading an existing license is very similar to the license distribution process. The primary difference is that a platform challenge is not performed because a valid, digitally signed license is presented to the license server.

Fig. 8 shows the steps in a method for upgrading an existing license. Steps 172-176 are identical to those defined above with respect to Fig. 5. At step 240, the client 30 submits a valid software license to the intermediate server 32.

At step 242 in Fig. 8, the intermediate server 32 determines whether the license has expired and/or is for an older version. Assuming it meets one of these conditions, the intermediate server automatically contacts the license server 28 and requests that the license be upgraded (step 244). The intermediate server passes the old license and the client's system ID to the license server 28.

The license server 28 validates the old license and extracts the license's ID, which is used as an index into the client assignment table 116 in the secure license store 112. The license server 28 examines the table 116 to determine whether an

1 upgrade is available (step 246). If so, the license server 28 upgrades a record in
2 the table, consuming one upgrade license, and returns an upgraded license to the
3 intermediate server 32 (step 248). The intermediate server 32 forwards the
4 upgraded license to the client and completes the connection (step 250). The client
5 30 replaces the old license with the upgraded one in the license cache 136 (step
6 252).

7 As a matter of policy, licenses are assumed to be backward compatible.
8 That is, a next generation 5.X license is always accepted by a current ^{generation} ~~generation~~
9 4.X server. This allows a customer to have a seamless mix of different servers.
10 Variances in the licenses internal data structures are taken into account by
11 including a version number within the license.

12 13 Temporary Licenses

14 Suppose a client 30 requests a software license, but the license server 28
15 does not have an available license in the secure license store. In this case, the
16 license server 28 issues a temporary license that is valid for a finite duration (e.g.,
17 60 days).

18 With reference to Fig. 3, the requesting client submits its system ID 142 to
19 the intermediate server 32, which forwards the client's system ID 142 to the
20 license server 28. The license server 28 generates a temporary license and
21 associates it with the client's system ID 142. The temporary license is passed back
22 through the intermediate server 32 to the client 30. Each time the client presents
23 the temporary license, a new license request is generated. Once the license server
24 has an available license (e.g., the license server purchased additional licenses from
25

